Отчет о создании “Сервиса счетчиков”.

Время создания: *25.08.2021*

1. Цель теста

Бизнес-цель теста – *получение навыков работы с распределенными транзакциями.*

Техническая цель теста – использование механизма захвата изменений данных с коммутативными обновлениями.
Нагрузочное тестирование платформы *Debezium*.

2. Описание решения

1. При реализации масштабируемой архитектуры сервиса счетчиков, с обеспечением консистентности между счетчиком и реальным числом непрочитанных сообщений, используется платформа *Debezium* с реализацией через *debezium-embedded*. Данное решение позволило реализовать транзакционную отправку сообщений через брокер *Kafka*.
2. Так же в самом микро-сервисе счетчиков реализован паттерн «Коммутативные обновления».
3. Для микро-сервиса счетчиков создана отдельная БД.

3. Функциональное описание

1. Для реализации решения было добавлено два микро-сервиса (см. *Схему 1*). Первый *cdc-producer* на основе *debezium-embedded* для транзакционной отправки сообщений. При конфигурации данный микро-сервис подключался в качестве slave к БД *MySQL*. В чтении репликации *MySQL* настроены таблицы *dialog_message* и *user*. Информация о прочитанных бинарных логах также сохраняется в топик *Kafka*. Декодированные логи, при помощи *debezium-embedded*, о сохранённых/измененных данных в этих таблицах в результате транзакций, отправляются в отдельный топик *Kafka* в виде *JSON* сообщений.
2. В микро-сервисе счетчиков из топика *Kafka* читаются *JSON* сообщения об изменениях в таблице *user* и заносятся в таблицу *user* БД данного микро-сервиса. Также для реализации паттерна «Коммутативные обновления» анализируются данные об изменениях в таблице *dialog_message* и при создании сообщения для пользователя инкрементируются счетчики *total* и *unread* в таблице *counter* а при изменении записи о сообщении на статус “прочитано” декрементируется счетчик *unread*.

Схема 1.

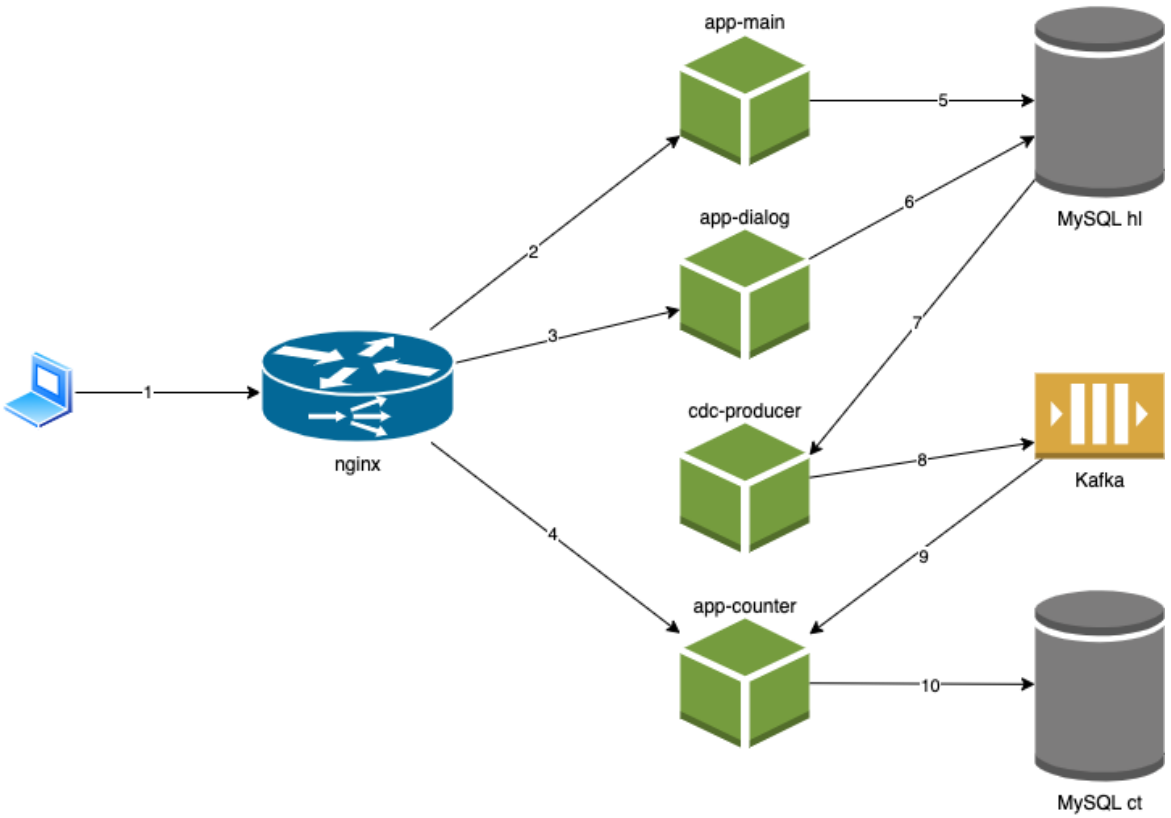

Таблица 1.

1	*Взаимодействие клиентского приложения React с API REST*
2	*API REST основного микросервиса*
3	*API REST микросервиса диалогов*
4	*API REST микросервиса счётчиков*
5	*Подключение основного микросервиса к БД*
6	*Подключение микросервиса диалогов к БД*
7	*Репликация основной БД на микросервис debezium*
8	*Сообщения о прошедших транзакциях в Kafka*
9	*Сообщения о прошедших транзакциях из Kafka*
10	*Подключение микросервиса счётчиков к БД*

4. НТ

1. Максимальная производительность системы – *143.37* запроса в секунду при 4-х потоковом тесте и *100* соединениях.
2. Время *отклика / выполнения* на уровне максимальной производительности по операциям составляло от *687.51* миллисекунд до *951.83* миллисекунд. Так же были просадки после *90 перцентиля* в диапазоне *1-2,45* секунд.

Диаграмма 1.

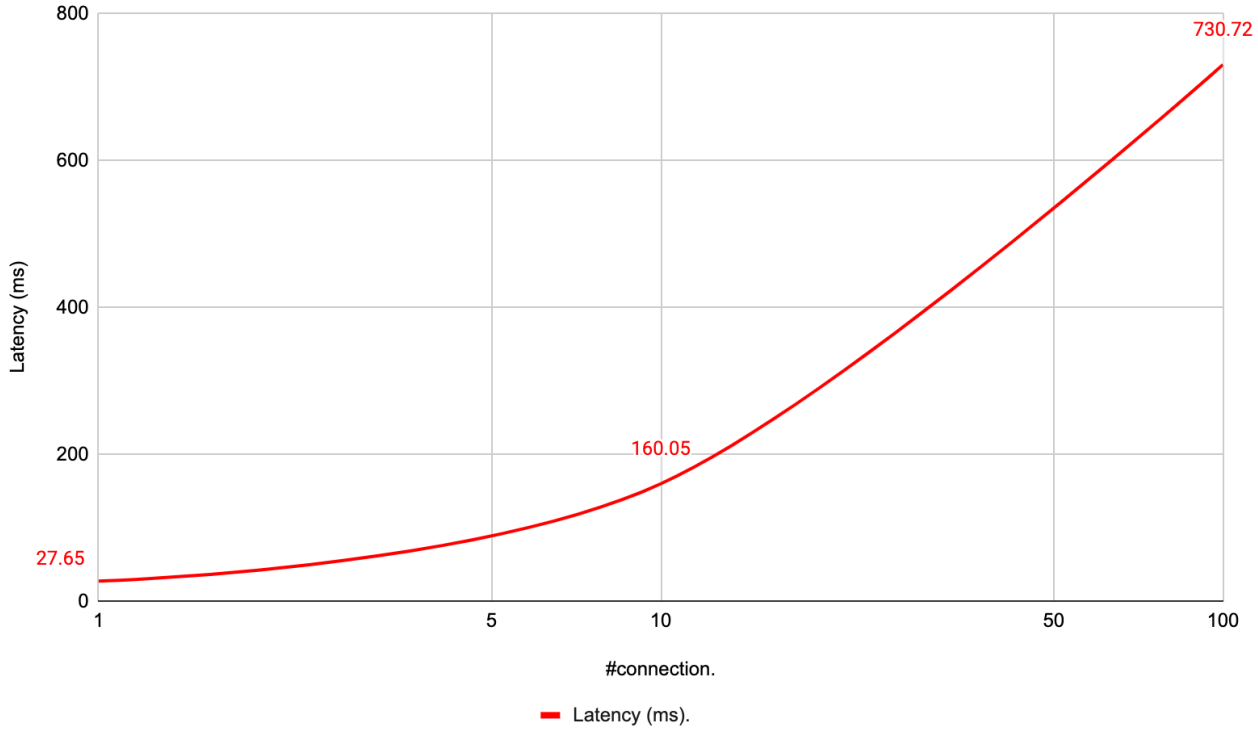

Диаграмма 2.